

APPENDIX B

Sample Code for Second Preferred Embodiment

```
/*=====
5   Method2 sample code:

   NOTE: This code is intended to demonstrate the key
   points of the implementation for method 2. It is
   intended for clarity and simplicity, so it has not
   been optimized.

10  The code is written in C with two C++ extensions:
   *   C++ style comments (everything from "//" to
       the end of the line is a comment)
   *   Variables can be declared anywhere in a
15  function, not just at the start of a scope.

   This code assumes:
   *   An OpenGL context to display the pixels has
       been created and is active
20  *   The GL renderer supports the
       GL_TEXTURE_RECTANGLE_EXT extension -
       non power of 2 pixels. The method would work
       without the extension, but would not be as
       optimal or simple.
25  *   The size of the out-of-order pixel data is
       stored in sPixelDataRect.
   *   The size of each pixel is stored in
       sBytesPerPixel. This code assumes a 2 or 4 byte
       pixel. The method will work with 1 byte pixels,
30  but the implementation is more complicated.
   *   A 2D texture, the same size as the out-of-order
       pixel data has been created and is bound to the
```

```
        id stored in sTextureID.
    *   A mask texture has been created and bound to
        the id stored in sMaskTextureID. This texture
        is one pixel high and two (for 4 bytes per
5       pixel out-of-order data) or four (for 2 bytes
        per pixel out-of-order data) pixels wide. The
        left most pixel is white and fully opaque. All
        the other pixels in the mask texture are fully
        transparent.
10
        Written by: Mick Foley (mickf)

        Copyright:      2003 Microsoft
        =====*/
15
        // header files that define the OpenGL data types,
        // values and functions

        #include <gl.h>      // the OpenGL header
20     #include <glxt.h>    // OpenGL extensions

        // local type definition

        typedef struct struct_tRect
25     {
            long fTop;
            long fLeft;
            long fBottom;
            long fRight;
30     } tRect;

        // static data - see the notes for more info
```

```
static void* sPixelData_BaseAddress;
static tRect sPixelDataRect;
static long sBytesPerPixel;
static int sTextureID;
5 static int sMaskTextureID;

// sub-routine declaration

static void
10 Method2_DrawRectWithOffsetAndMask(
    long inOffset,
    long inColumn );

// the code
15 void Method2_DrawPixels( void )
{
    // set up the two textures...
    glActiveTexture( GL_TEXTURE0 );
    20 glEnable( GL_TEXTURE_2D );
    glEnable( GL_TEXTURE_RECTANGLE_EXT );
    glBindTexture( sTextureID );
    glTexEnvi( GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,
        GL_DECAL );
    25
    glActiveTexture( GL_TEXTURE1 );
    glEnable( GL_TEXTURE_2D );
    glBindTexture( sMaskTextureID );
    glTexEnvi( GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,
    30 GL_MODULATE );
    glTexParameteri( GL_TEXTURE_WRAP_S, GL_REPEAT );
    glTexParameteri( GL_TEXTURE_WRAP_T, GL_REPEAT );
```

```
// update the texture with the data
// from the emulator VRAM
if ( sBytesPerPixel == 2 )
{
5      glTexSubImage2D( GL_TEXTURE_RECTANGLE_EXT,
        0,
        sPixelDataRect.fLeft,
        sPixelDataRect.fTop,
        sPixelDataRect.fRight -
10      sPixelDataRect.fLeft,
        sPixelDataRect.fBottom -
        sPixelDataRect.fTop,
        GL_RGB,
        GL_UNSIGNED_SHORT_5_6_5,
15      sPixelData_BaseAddress );
    }
else
{
    glTexSubImage2D(GL_TEXTURE_RECTANGLE_EXT,
20      0,
        sPixelDataRect.fLeft,
        sPixelDataRect.fTop,
        sPixelDataRect.fRight -
        sPixelDataRect.fLeft,
25      sPixelDataRect.fBottom -
        sPixelDataRect.fRight,
        GL_BGRA,
        GL_UNSIGNED_INT_8_8_8_8_REV,
        sPixelData_BaseAddress);
30      }
}
```

```
// draw the rectangles
```

```
if ( bytesPerPixel == 2 )
{
    // draw four rectangles with
    // different columns and offsets
5
    // draw column 0, offset right by 3
    Method2_DrawRectWithOffsetAndMask( 3, 0 );

    // draw column 1, offset right by 1
10    Method2_DrawRectWithOffsetAndMask( 1, 1 );

    // draw column 2, offset left by 1
    Method2_DrawRectWithOffsetAndMask( -1, 2 );

    // draw column 3, offset left by 3
15    Method2_DrawRectWithOffsetAndMask( -3, 3 );
}
else
{
20    // draw two rectangles with
    // different columns and offsets

    // draw column 0, offset right by 1
    Method2_DrawRectWithOffsetAndMask( 1, 0 );
25

    // draw column 1, offset left by 1
    Method2_DrawRectWithOffsetAndMask( -1, 1 );
}

30 // finished with all the commands
glFlush();
}
```

```
void
Method2_DrawRectWithOffsetAndMask(
    long inOffset,
    long inColumn )
5  {
    // while the pixel texture's texture coords are in
    // source pixel increments, the mask texture uses
    // GL's more common 0.0 - 1.0 mapping, so we need to
    // convert (this is needed because the extension
10  // that allows pixel level text coordinates,
    // GL_TEXTURE_RECTANGLE_EXT, does not allow repeat
    // modes for drawing)

    GL_FLOAT maskTextureScaleFactor;
15  GL_FLOAT maskTextureLeftCoord;
    GL_FLOAT maskTextureRightCoord;

    if ( sBytesPerPixel == 2 )
    {
20      maskTextureScaleFactor = 0.25;
    }
    else
    {
        maskTextureScaleFactor = 0.5;
25  }
    maskTextureLeftCoord = maskTextureScaleFactor
        * ( ( GL_FLOAT )inColumn );
    maskTextureRightCoord =
        ( ( GL_FLOAT )( sPixelDataRect.fRight
30      - sPixelDataRect.fLeft )
        + ( ( GL_FLOAT )inColumn ) )
    * maskTextureScaleFactor;
```

```
// prepare to issue the draw commands
glBegin( GL_QUADS );

// upper left vertex
5 glMultiTexCoord2i( GL_TEXTURE0,
    sPixelDataRect.fLeft, sPixelDataRect.fTop );
glMultiTexCoord2f( GL_TEXTURE1,
    maskTextureLeftCoord, 0.0 );
glVertex2i( sPixelDataRect.fLeft + inOffset,
10    sPixelDataRect.fTop );

// upper right vertex
glMultiTexCoord2i( GL_TEXTURE0,
    sPixelDataRect.fRight, sPixelDataRect.fTop );
15 glMultiTexCoord2f( GL_TEXTURE1,
    maskTextureRightCoord, 0.0 );
glVertex2i( sPixelDataRect.fRight + inOffset,
    sPixelDataRect.fTop );

20 // lower right vertex
glMultiTexCoord2i( GL_TEXTURE0,
    sPixelDataRect.fRight,
    sPixelDataRect.fBottom );
glMultiTexCoord2f( GL_TEXTURE1,
25    maskTextureLeftCoord, 1.0 );
glVertex2i( sPixelDataRect.fRight + inOffset,
    sPixelDataRect.fBottom );

// lower left vertex
30 glMultiTexCoord2i( GL_TEXTURE0,
    sPixelDataRect.fLeft, sPixelDataRect.fBottom );
glMultiTexCoord2f( GL_TEXTURE1,
    maskTextureRightCoord, 1.0 );
```

```
glVertex2i( sPixelDataRect.fLeft + inOffset,  
            sPixelDataRect.fBottom );  
  
// done with the rectangle draw  
5 glEnd();  
}
```